Dependable Wireless Sensor Networks

Françoise Sailhan* — **Thierry Delot**** — **Animesh Pathak***** — **Aymeric Puech****** — **Matthieu Roy*******

* *CNAM, 292 rue Saint-Martin, F-75141 Paris. firstName.surname@cnam.fr*

** *Université de Valenciennes, Bat. ISTV2, Le Mont Houy 59313 Valenciennes. firstName.surname@univ-valenciennes.fr*

*** *Projet Arles, INRIA Rocquencourt, domaine de Voluceau, BP 105, Le Chesnay. firstName.surname@inria.fr*

**** *SmartGrains, Paris Innovation Bourse, 5 rue d'Uzès, 75002 Paris. firstName.surname@smartgrains.fr*

***** *TSF Group, CNRS ; LAAS ; Université de Toulouse ; UPS , INSA , INP, ISAE ; 7 avenue du Colonel Roche, F-31077 Toulouse, France. surname@laas.fr*

RÉSUMÉ. *L'explosion de l'instrumentation de notre environnement conduit au besoin de développer des réseaux de capteurs qui non seulement sont capable d'observer et de réagir mais qui aussi témoignent d'un niveau de tolérance aux fautes élevé. A cette fin, nous indentifions les fautes critiques dont sont sujets ces réseaux, puis, nous discernons les principaux défis liés au développement de réseaux de capteurs robustes, et cela, en nous réferrant à deux cas d'étude, un système de surveillance et un système de géo-localisation. Enfin, nous envisageons le design d'un système d'injection de fautes qui vise à insérer délibéremment des fautes dans le réseau de capteurs opérationnel de façon à déterminer si la cible réagit de façon adéquate.*

ABSTRACT. *The explosion in the instrumentation of our environment is driving the need for developing sensor networks that not only observe and measure but that also show a high-level of fault tolerance. Towards this goal, we identify faults and key challenges related to the development and deployment of self-managing dependable sensor networks, on the account of two use cases: a monitoring system and a geo-localization application. We then cater requirements and envisage the design of a fault injector which deliberately injects software faults into an operational sensor network in order to determine if the target offers an effective response.*

MOTS-CLÉS : *Réseau de capteurs sans fil, sureté de fontionnement, injection de fautes.*

KEYWORDS: *Wireless sensor networks, fault injection, dependability.*

1. Introduction

Wireless sensor networks have been founded with the fundamental aim of monitoring a phenomenon (sensing) and performing actions in response (actuation). The architectural premise of such networks is a set of networked sensors, embedding a detector and an electromechanical (actuating) device, being together capable of observing, measuring, reacting and communicating on a phenomenon and sharing it with each others. To get useful observations and provide an adequate response, several detectors and actuators may be required, each providing a basic observation and the corresponding reaction (e.g., a temperature sensor and a heating element), which may be combined to provide a complete behavior. With sensors, the key constraint remains resource saving. It has served as the primary argument for developing specific energy-, memory-, computationally-aware, low-data-rate protocols, systems and applications that together attempt to extend nodes lifetime. However, as the range of applications extends in the fields of industrial, medical and mission-critical systems, additional requirements related to robustness and reliability must be taken into account (Barrenetxea *et al.*, 2008). Currently, we are confronted with wireless sensor networks that are optimized for their gathering abilities but that fail at ensuring their own reliance and their proper operation. Such assertion is reinforced by the increasing number of real-life deployment campaigns that fail (Barrenetxea *et al.*, 2008).

Meeting the challenging task of developing dependable sensor networks necessitates not only to provide fault-tolerant sensing and actuating capabilities but also in order to evaluate and validate their dependability attributes. Towards this goal, we present the foundations of a fault injection-based evaluator that deliberately accelerates the occurrence of faults to evaluate the quality of error handling mechanisms and, more generally, to analyze the dependability of the sensor network. This solution is grounded on two main building blocks, namely :

– a *fault injection mechanism* that inserts faults into one (or several) run-time node(s) in a remote and distributed way,

– a *monitoring component* that checks for any misbehavior or dysfunction at runtime, and reports it.

These components together aim at evaluating the dependability of the applications and protocols deployed over a sensor network. In order to demonstrate the effectiveness of the proposed fault injector, we envisage two specific use cases : a fire fighter tracking system and a parking lot occupancy monitoring system.

The remainder of this paper is organized as follows. We first introduce preliminary concepts, describing for this purpose wireless sensor networks, failures and surveying existing literature on dependability evaluation (§2). Grounded upon this survey, we emphasize the key challenges related to designing, implementing and evaluating the dependability of self-manageable sensor networks, focusing on two use cases, a parking lot monitoring system and a geo-localization system for first aid response. Then, we propose an approach for assessing the dependability of a sensor network by injec-

ting faults, focusing more particularly on the communication layer (§3). Finally, we conclude this article with two variants of a case study (§4).

2. Preliminaries

In our view, dependability evaluation of a sensor network follows a three-steps-process. First, one has to analyze the purpose- and application-dependent characteristics of sensors as an attempt to model wireless sensors (§2.1). Grounded upon this understanding, one can identify the critical failures that threaten a sensor network (§2.2). Finally, taking into account the traditional approaches for evaluating dependability (§2.3), one can envisage and suggest a wireless sensor-oriented approach for assessing dependability. But, before proceeding further, let us backup a little and define what is a wireless sensor network, abstracting and modeling for this purpose sensors, actuators, and their relative programming models.

2.1. *Wireless Sensor Network*

Generally speaking, the objective of a wireless sensor network is to observe, measure, communicate and possibly react to a phenomenon. Phenomena can take multiple forms, e.g., a physical property (temperature), a classification (e.g., species), frequency, count or simply an existence indication. An observation is an event with a result that describes the above phenomenon ; such value is expressed with regards to a reference system which provides the context for the observation interpretation. To get useful observations, several detectors may be required. A sensor is usually built using one or many detectors ; each provides a basic observation that may be combined (i.e., aggregated into collections using composite patterns) with other classifiers to get a useful observation. One of the many attempts (Girolami *et al.*, 2008) to provide a model of such a sensor is materialized by the sensor model language (Boots *et al.*, 2007) and the observations and measures model (Cox, 2007) which together present the conceptual model and the related XML-encoding framework[1] for observation and measurement together. In addition to the sensing component which is intended to observe the phenomenon, a sensor also consists of a computation, a storage, a wireless-radio communication subsystem, a power supply and possibly a actuating device. It follows that attempts to model sensors distinguish the virtual sensor (that performs observation and provides the relative measurements, as described above) from the hardware sensor (i.e., physical sensors, as described bellow). Such a low-level modeling is mostly focused on abstracting hardware and allowing flexible control on nodes as it is the case with existing wireless sensor network operating

1. Note that these normative reference documents do not make recommendations about the implementation of the fundamental definitions ; different XML-based standards being usable. Instead, they provide a common framework for describing virtually sensor system, focusing on the description of the meaning of the phenomenon and its representation.

systems (e.g., Contiki (Dunkels *et al.*, 2004)) extended with stack-oriented virtual machine (e.g., ASVM (Levis *et al.*, 2005)) which all provide abstractions along with related mechanisms. A complementary attempt to model a sensor is materialized with the Management Information Base (MIB for short) (Kim *et al.*, 2010) defined to operate in conjunction with the Simple Network Management Protocol (SNMP) (Mukhtar *et al.*, 2009). The proposed model is mostly focused on communication. Indeed, apart from information on energy (e.g., the power source, ability to receive power from the alternating current mains, ability to conserve power), other parameters described as part of the MIB include the name of the supported routing protocol, the related routing tables and the ability to allocate address. The same applies to the other models proposed in the literature (Mourad *et al.*, 2008) which are network-centric rather than being platform-centric ; the sensor network being viewed as a distributed system (in practice, a graph) organized based on several criteria including e.g., neighboring, logical groups. The basic idea here being to promote the use of localized algorithms. This separation between platform-centric and network-centric models mainly comes from the programming platform that have been devised. Basically, two major classes of applications, data collection-type and collaborative information processing-type, lead to the creation of two levels of abstractions that ease the programming of distributed applications. They refer to :

– the so-called macro-programming which provides a language construct that handles multiple nodes collectively and a set of operations to perform on it, thus enabling to easily program collaborative tasks,

– in-network processing which constitutes a database-oriented that provides an intuitive way of accessing the sensor information by hiding the in-network summarization and aggregation.

Initiatives to ease the programming of sensors are naturally not limited to macro-programming and in-network processing but are extended, as the research progresses on the field, with alternative paradigms e.g., component-oriented platforms (Janakiram *et al.*, 2005) or spatial programming wherein resources are identified and referenced based on their physical location (Borcea *et al.*, 2004). The aforementioned programming models accommodate failures by making the programming model adaptive by e.g., by providing run-time binding or dynamic task reassignment ; such adaptivity being out of the control of the user of application programmer.

Overall, it significates that faults/errors/failures are masked to application developper (in opposition to being showed), which renders dfficult the identification/localisation of faults. In practice, it also means that error handling is not programmed neither at the node level or network level by the user or application developer.

2.2. *Faults*

In the literature, several deployment campaigns of wireless sensor networks within outdoor environment including potato fields (Thelen *et al.*, 2005), or high-mountain

sites (Barrenetxea *et al.*, 2008), have been reported. Oscillating between failures and non-trivial but successful deployments, those campaigns demonstrate the difficulty of effective deployment of WSN in the real world,compared to simulation. They also emphasize the need for monitoring tools that assess the health of the network as well as deployment-time tools that ensure the system is up and running once it is deployed. Such requirements can be summarized (Barrenetxea *et al.*, 2008) by the simple-but-meaningfully recommendation : « Don't be a black box. Keep in mind that programming embedded devices requires a different philosophy than traditional programming ». Such deployment campaigns provide many information on the specific type of faults found in sensor networks. To date, errors that have been reported in the literature come from both hardware faults and software faults. For instance electronic circuits are subject to two main classes of faults :

– transient faults implicate that the sensor recovers its normal behavior when e.g., the system is reset or the fault stimulus ceases,

– permanent faults inflect defects that have a permanently effect.

One may find many similarities between computer-related faults and sensor-related faults : sensors, built as a complex combination of hardware and software, are subject to classical human-related faults, e.g. , software bugs, memory leaks, memory corruptions and pointer-initiated memory violation (Chen *et al.*, 2009). Nevertheless, sensors networks differentiate themselves by the amplitude of *natural faults* which define the faults that that are caused by a natural phenomenon. This type of faults, which span both hardware and software, remains mainly **unforeseen** : errors are discovered on the field during, or after, the deployment campaigns. In practice, either external (e.g., water infiltration) or internal natural processes (e.g., power transient) cause physical deteriorations. Notice that we have herein no pretention of neither documenting the state-of-the-art on sensor deployment campaign failures or providing a full coverage of potential failures. Recall that, while being too rarely reported in the literature, failures are application- and mission-dependent. Illustrating example of natural failures include :

– Direct sunlight that swamps the sensor infrared signal (Milla *et al.*, 2006).

– Degradation of the battery. Note that this latter can be simulated as an incipient change in a battery capacitance.

– Different temperature responses in the processor and radio oscillator, which causes numerous network failures (Barrenetxea *et al.*, 2008).

– Water infiltrations, which introduce degradations within the hardware (Milla *et al.*, 2006).

Moreover, sensors are usually mass produced with cost reduction in mind, and the use of off-the-shelf components in their design aggravates the vulnerabilities of sensors. Possible faults are obviously not limited to mechanical systems. Instead, they also cover communication faults (e.g., multi-path fading, noises) that are inherent to wireless radio signaling. It is worth noting that communication faults are amplified by multi-

hops communication mechanisms and tend to extend to the gathering layer wherein massive corruption on the sensed data has been reported in the early-age literature.

It is a common place to argue that a fault do not always manifests itself. A fault may be active (i.e., causing an error) or dormant. It can also be masked unless this latter reaches the service interface, leading to a service failure. With sensor networks, failures are often revealed at the fusion center (sink) wherein the end user makes decisions on the collected data quality. This data-centric manifestation of failures may be caused by the deterioration/corrosion of the sensing element, or by any of the aforementioned hardware-, software-, communication-type of faults. These failures have been pointed out and categorized as follows :

– *Outliers* : a single isolated event that is outside the expected range of values to be returned. Outliers are sub-categorized as short simple or long segmental (Ni *et al.*, 2009). *Short simple outliers* are high frequency noise/error usually represented as abnormal sudden bursts and depressions, which show great dissimilarity with other part of the same sensing series. Instead, long segmental outliers refer to erroneous sensed readings that cannot reflect the environmental change and that last for a certain time period.

– *Stuck-at faults* : A series of data values with little or no variation for a period of time longer than expected : data is frozen and remain to a given value (e.g., top of roof),

– *Spikes* : A change in gradient over a period of time much greater than expected,

– *Abrupt fault* : It applies a constant offset to the value.

– *Incipient fault* : An offset that starts at zero and increases linearly (i.e., slowly developing) is added to the value,

– *Excessive Noise* : Data exhibits much higher noise than expected, but may still track the phenomenon.

Depending on the degree of severity, collected data is either totally useless or still provides some useful information about the phenomenon of interest. In order to overcome the above enumerated deficiencies, various approaches have be put together.

2.3. *Approaches for Overcoming Deficiencies*

Developing reliable and fault tolerant hardware and software together is a particularly challenging task mainly because it requires discipline during any of the design and implementation phases. A wide range of strategies toward dependability, involve prevention, detection, reaction, self-configuration, self-healing, defensive design, development or maintenance, deep experience and best engineering and operational practice. Beside these design- and implementation-time tacticals and strategies, which basically are intended to foresee and overcome operation-time run-time issues, approaches for developing highly reliable systems are put in a concrete form through rigorous testing or/and the use of formal modeling-based dependability evaluation.

With this latter, the expected behavior of the studied system is constructed and compared against the observed behavior ; in this view, any deviation is interpreted as a fault. Another popular approach lies in modeling the behavior of the system on a given fault state ; the fault related to a particular model is considered as detected when this above prediction model is sufficiently close to the observed model. With the two above cases, various formal techniques e.g., classic process algebras (Fehnker *et al.*, 2007), timed automata (Dong *et al.*, 2008), model checking (Fehnker *et al.*, 2007; Demaille *et al.*, 2006), have been respectively used to ease the analysis of sensing-actuator networks, bio-medical sensor, the so-called LMAC protocol (van Hoesel *et al.*, 2004) and a sensing application (Demaille *et al.*, 2006). Regardless of the method, authors emphasis on the impossibility to prove correctness in the general case (i.e., for any kind of network topology) and the difficulty to model the system due to the complex behaviors implied by the distribution of sensor nodes.

Hence, due to the inherent difficulty of deploying sensor networks, usual dependability measurements for sensor networks are performed on simulators. The main dependability attribute that is measured is the availability of the service (to be more precise, its expected availability). Cyber Physical Systems are meant to interact with the environment, and thus provide information and measurements on continuous physical parameters, like localization or temperature. In such systems, dependability can be evaluated on more precise attributes, e.g., the accuracy of the service, or even the expected accuracy given a particular distribution of failures. More importantly, diagnosis of such systems can take advantage of the interaction that the system has with its environment by providing information on physical weaknesses of the configuration, such as areas where sensors should be duplicated. Indeed, the strong relationship between the physical world and the sensor network should be made apparent in the dependability evaluation.

An alternative attempt for evaluating the dependability of a sensor network lies in performing fault-injection experiments. This approach aims at accelerating the occurrence of faults for the purpose of assessing the effectiveness of built-in detection and recovery mechanisms. Faults are injected either at the hardware level – involving for this purpose e.g., logical, electrical faults or electromagnetic interferences – or at the software level in which case the code and data are corrupted. Physical fault injection requires an access to the targeted platform and takes places during the testing and integration phases. In contrast, simulation-based techniques are used during the design and implementation phase. In practice, software implemented fault injection (SWIFI for short) makes use of additional software in order to inject faults in the physical system. This provides a cost efficient and flexible way of injecting faults. Software fault injection implies reproductible, targeted system stimulation and permits to verify (software/hardware) fault tolerance, reproduce easily faults events to debug fault tolerant mechanisms, validate software operation over a full input range variance, noise and parametrized drifts. To the best of our knowledge, fault injection in wireless sensor networks has been limited to software fault injection in a single sensor (Barrenetxea *et al.*, 2009). The approach emulates hardware faults at the assembly level, flipping bits in the code/data memory as well as processor registers. It follows that developers

of application and protocols for wireless sensor network are still being forced to develop *ad hoc* testing approaches to validate the dependability of their applications and protocols.

3. MURPHY : Dependability Evaluation in Wireless Sensor Networks

The absence of tools for facilitating and validating the dependability of wireless sensor networks circumvents the need for developing a fault injector that can scope with the networking and distributed nature of WSN applications and protocols. Note that existing communication fault injectors that have been introduced to deal with computers network, e.g., Orchestra (Dawson *et al.*, 1996) or ComFIRM (Jung-Drebes *et al.*, 2006), are not appropriate to test and validate wireless sensor networks because they are specific to proprietary technologies and particular computer operating systems, which excludes their portability on resource-less OS-specific sensor nodes (Jung-Drebes *et al.*, 2006). This calls for developing embedded and low resource-consuming fault injection that scopes with the networking and distributed nature of WSN resource-constraint applications and protocols.

As a first step upon that goal, we envisage two case studies (§3.1) a parking lot monitoring system and a context-aware geo-localization system. Looking one step further, we introduce the Murphy project, which aims at integrating the last advances in terms of work flow compute-actuate technologies so as to support the distributed, automatic, and seamless generation and insertion of faults in order to monitor and diagnose a given sensor network.

3.1. *Use Cases*

We are envisaging two case studies, a parking lot monitoring system and a context-aware geo-localization system, which imply distinct scenarios (a urban transportation planning for the former and an emergency/sinister scenario for the latter), which in turn induce different types of impairments (faults, errors and failures) and dependability requirements.

Urban transportation planning. Wireless sensor networks show a great potential for designing new arrays of large-scale real-time parking lot monitoring applications. In practice, an operator positions wireless sensors on each parking lot and a cooperative software makes these latter cooperate with one another in order to obtain a real-time map of parking occupancy. This information allows drivers to park their car faster (thus improving the QoS perceived by visiting drivers), while permitting car park managers to improve their processes and resource allocation (e.g. locally lowering the mechanical ventilation systems). Similarly, large cities on-street parking solutions could benefit from this sensor-driven application by significantly reducing the share of traffic cruising (i.e., cars that are waiting for curb vacancies), which accounts for one third to one half of car gas emissions.

Faults are encountered at any stage of WSN development, from the high-level design of embedded software down to hardware faults, energy failures, or harsh environmental conditions. The fault instantiation may take various shapes, such as partial / complete shut-down of the network, missing network frames or corrupted data (deprecated outputs, unusable outputs). Such an application poses challenges related to (1) guaranteeing the robustness of the underlying wireless application-led sensor network under harsh conditions, (2) evaluating the level of accuracy of the gathered information, and (3) validating the fault tolerant and automatic fine-grained recalibration mechanisms that have been implemented.

Context-aware geo-localization. Another example of WSN application stems from emergency and disaster scenarios. As illustration, the protection of art-crafts currently constitutes a major concern for the CNAM Museum (in French, Musée des Arts et Métiers[2]). Indeed, an on-going preventive project (Sailhan *et al.*, 2009) is being carried out ; the objective of this project is to provide sensor-assisted monitoring and tracking of both art-crafts and fire-fighters during a devastating fire. In practice, this involves gathering environmental information (at least temperature and humidity) to evaluate the disaster progress while keeping track of art-crafts and fire fighters to facilitate their rescue whenever needed. This tracking system assigns geographic coordinates to every node in the system (a node is an art-craft piece or a firefighter). For this purpose, wireless sensor motes, herein simply called sensors, are deployed over the geographical area so as to play the role of anchors, i.e., nodes which are aware of their own location. Apart from furnishing geo-localization information, sensors provide information (temperature, humidity), which enables the geo-localization system to adapt to varying environment conditions that may affect the localization accuracy. In practice, geo-localization follows a two-steps process. First, a pairwise distance (between the node and an anchor) is measured. To that means, quantitative ranging (based on the signal strength, the time to flight and the angle of arrival) is correlated with a model of path loss so as to estimate the distance and angle separating the anchor to the node. Second, several distances and angles are combined using multi-lateration/triangulation so as to estimate the node position. In order to improve the geo-localization accuracy, redundant measurements (step 1) and estimates (step 2), are filtered based on various statistical methods, e.g., mean, standard deviation[3]. Notice that the precision of the geo-localization may be further improved based on sensed information (temperature, humidity).

As in the first use case, *faults* may occur during any of the above steps. Beside the monitoring-specific faults (as described in the last paragraph), we distinguish three additional categories of faults that impact the signal quality. As such, they are specific to geo-localization and communication. We categorize these faults as basic faults (i.e., the faults that are commonly encountered in both indoor and outdoor environments), faults that are specific to indoor operation and faults specific to the occurrence of a si-

2. `http://www.arts-et-metiers.net`
3. Alternative approaches include probabilistic methods, e.g., family of Kalman filters, particle filters, or neural networks.

nister. **Classical geo-localization errors** mainly come from *line-of-sight radio fading factors* (e.g., multi paths, beam spreading, signal distortion) and accidental/intentional interferences (electrical equipment), on or surrounding the operational frequency. In addition to the above, **indoor operation** faults imply the presence of obstacles and spatial irregularities that cause static or time varying errors. *Static errors* result from (i) the physical arrangement of the objects and/or (ii) the building frame. As such, they are constant over the time and may be overcome thanks to finger-printing. In contrast, time varying errors are unpredictable and are thus usually modeled as random. Note that to overcome the effect of errors, peer-wise measurements are typically repeated over a short period of time, averaged and filtered. Finally, a sinister context introduces **climatic varying conditions** e.g., varying degrees of temperature, humidity, luminosity, smoke, rain drops, sand, dust and flames. Precisely, *hydro-meteors* (i.e., water drops, vapor) cause dominant fading in the micro-wave whereas *solid particles* which are suspended in the air (i.e., sand, smoke) should be taken into account to a minor extent. In addition, *microwave thermal noise* is introduced while *multi-paths* effects are exacerbated by water surfaces, wich constitute strong reflectors. Depending on the severity of the sinister, the above climatic conditions may lead to minor up to major perturbation of the geo-localization system (perturbated ranging and triangulation/multi-lateration) along with the communication system(link failures). In addition, the sinister is a major factor of destructions/injuries that may lead to the (partial versus complete) destruction, and thus unavailability, of the anchor(s). Whenever undetected, the above faults may cause error(s), which in turn may lead to failure(s) unless the monitoring and/or geo-localization system is capable of handling it. It is therefore critical to both analyze the impairments that may affect dependability so as to define means for reducing and preventing their effect.

3.2. *Dependability Analysis*

A failure may imply different degrees of severity that range from minor up to catastrophic. For the sake of clarity, we attempt to classify the impact of failures according to the use cases (transportation planning versus emergency scenario ; monitoring versus geo-localization) even if such failures are not mutually exclusive.

Parking lot Monitoring. Providing a dependable service constitutes clearly the main objective of a parking lot application. Among potential attributes of dependability[4], the attributes of primary importance include :

– *Availability*. Availability represents a major property. It is clearly affected by the remaining energy on a sensor mote and its ability to self-recover from any failure. Means for improving reliability include developing low energy-consuming protocols, preventing communication (communication being the major source of energy consumption), relying on wireless mesh backbone wherein a set of power-lined sen-

4. Note that our attempt herein is not to provide a definition of dependability and its attributes, interested reader may refer to (Avizienis *et al.*, 2008)).

sors takes in charge multi-hops routing, and ensuring self-healing mesh flushing, booting, configuring, and routing,

– *Integrity*. Integrity is mostly focused on collected data integrity and quality. Note that data integrity is not only a matter of networking or storage. Data integrity can be traced back to the calibration of the sensor, the harsh (natural) conditions, up to wireless transfer/retrieval alteration or destruction.

Geo-localization. Disaster relief and emergency management implies two dependability properties, namely safety and survivability.

– *Safety*, the state of being safe, refers to the ability of being protected against the consequences of geo-localization errors that may harm a fire fighter. As such, safety constitutes the primary concern of the application. We distinguish two sub-categories of threats, localization precision (to a minor extent) and localization accuracy (to a major extent), that impact the safety :

- The localization *precision*, which is the level of granularity of the estimation that is provided by the geo-localization system, is to be ameliorated by the context-awareness of the geo-localization system. Such awareness is provided by a sensing and monitoring component. However, providing too high precision is not relevant : given the variability of the environment, and the fact that geo-localization is a guidance for humans, an estimate of about a couple of meters is sufficient.

- The geo-localization *accuracy* is subject to critical failures that come from false positives and false negatives. Supposing that the euclidean coordinates of a firefighter are given by $A = (x_A, y_A, z_A)$, a *false positive* consists in geo-localizing a firefighter at $B = (x_B, y_B, z_B)$ with the measure $||A - B|| >> \epsilon$, with ϵ defining the geo-localization precision. In other words, a false positive corresponds to the fact that a firefighter is located in a place where he is not. On the contrary, a *false negative* corresponds to denying that the firefighter is localized at A whereas the firefighter is actually located in A. False positive and false negative constitute two undesired fault that may lead to catastrophic consequences.

The two above factors, precision (to a minor extend) and accuracy (to a major extend) impact safety. Assuming that precision- and accuracy-related threats can be entirely prevented, constitutes a harmfull shortcoming. This circumvents the need for indicating the level/degree of confidence (i.e., accuracy and precision) that can be provided with regards to the sinister progress. In practice, this consists in displaying to the end users the degree of confidence in the coordinates that are provided by the geo-localization system.

– *Survivability* represents the ability that the system has to survive to the failures during a time scale bounded by the sinister duration. Basic principles that are applied during the design and development phase of the geo-localization system in order to increase survivability include providing decentralization (avoid a single point of failure), sensor redundancy and sensor robustness (based on adequate packaging). One fundamental that cannot be envisaged here is to place anchors outside the expected scope of damage : anchors should compulsorily be placed within the radius damage

so as to provide information that is necessary for geo-localization. Thus, destruction of anchor nodes progresses as the sinister evolves, rendering the anchors unavailable. In this study, (hardware) anchor survivability is not a primary concern : extending the lifetime of sensors by providing robust packaging, constitutes an issue that is addressed within the project. However, on the account of hight temperature, sensor destruction can be slowed down but cannot be prevented. Requirement is locked to ensuring a monitoring-enabled, fail-verbose (in opposition to fail silent or fail babbling) geo-localization system. Such anchor failure detection is critical to prevent the geo-localisation system from fault positive and fault negative. From the user point of view, fail-controlled geo-localization is restricted to clearly stating whereby geo-localization is alive, or cannot anymore be offered. Overall, the survavility objective is treefold. First, it consists in forcing the system to survive without direct human assistance (i.e., repair or placement of new anchors). Second, it requires to first circumsbribe the area where localisation can still or cannot anymore take place and then to provide this information to firefighters. Third, it entails to guaranty that it cannot fail in a way that can violate the safety property.

So far, we have provided two lists of impairments, one is common to most sensor applications (§2.2) whereas the second, which is described above, is focused on two case studies, monitoring and geo-localization. Whereas parking lot monitoring should exhibit a high-level of availability and data integrity, a geo-localization system intended to operate upon a disaster, should not violate safety and survivability properties. Although not aforementioned, data integrity remains a property of the geo-localization system (recall that this latter embodies a sensing and monitoring component). Regardless of the use case, the surveyed failures exhibit different degrees of harmfulness that range from minor to catastrophic. In order to prevent such failures, different means are applied, including (i) fault preventive measures that refrain the occurrence of fault forehand by designing and developing low-resource consuming protocols and applications, or by relying on mains-powered sensors, (ii) designing and developing fault tolerant mechanisms, avoiding the present of a single point of failure, providing autonomy, indicating the level of confidence on provided information, strictly defining operational/non-operational components) and (iii) forecasting faults, as described above. Despite the variety of means intended to avoid or overstep the occurrence of failures, testing still remains the only truly effective means for ensuring that the aforementioned properties are not violated. Thus, rather than referring to the process of finding bugs, a valuable testing mechanism must have a high probability of finding a yet not discovered error whose consequence may violate one of the dependability properties of the use case. Unfortunately, testing is a major time, resource and effort consumer, especially when safety is required. Our objective is to ease the development of dependable wireless sensor networks, through the extensive support for (i) distributed fault injection and (ii) monitoring and evaluation of dependability properties into a live wireless sensor network.

3.3. *Fault Insertion-based Dependability Evaluation*

Fault injection consists in accelerating the occurrence of faults for the purpose of easing the testing, benchmarking and assessment of dependability-related properties.Fault injection and dependability validation remains a complex process covering many of the activities performed during the development and deployment of both the system under study and the fault injector. These activities include specifying faults, developing and deploying a fault injector, defining a fault scenario to be used during testing campaigns. At any of these steps, one should envisage to answer at least to of the three following questions :

Which fault should be inserted and how to proceed ? This requires to specify the faults that are relevant to the use case under study. We believe that networking faults (i.e., signaling and communication faults), message- and data-oriented faults, as well as network covering issues and faults focusing on the distributed nature of applications are the most relevant class of faults simply because applications and protocols deployed across wireless sensor network are by nature designed with autonomy, fault tolerance et hence distribution in mind. This fault definition process is closely related to the specification and implementation of the injector of that fault. Example of communication faults thus include message, packet, data manipulation (modification/corruption), as well as their removal or addition (generation of workload is considered as a key instrument for analyzing the network capacity). Among the above operations, user should select one operation to be performed, and possibility combine one with each other to get a useful fault pattern. Overall, since a failure may result from a combination of faults, it follows that a fault pattern may involve one or several nodes.

Where should the specified fault and related code be injected ? To answer this question, one needs to rely on a convenient abstraction of the network to be able to reason on a given set of nodes (such that gateways of the sensor network, cluster heads, neighbors of a given node, 10 $\%$ of the total amount of nodes) wherein one or several fault should be injected. This underlying abstraction must be associated to a language and a related middleware to ease the work of the developer when this latter designates where the distributed fault should be injected at run time. Furthermore, in order to increase the performance and usability of the proposed fault insertion and dependability assessment system, we envisage to provide dynamic reassigned management responsibilities and task delegation to nodes as the topology evolves, hence providing seamless integrated fault injection and evaluation. This requires extending actual state-of-the-art macro programming technologies that have been successfully applied for sensor-actuate-compute sensor to support the seamless deployment and management of Murphy fault injector. These in-network query processing and mobile execution extra-capabilities will bring dependability evaluation in wireless sensor networks from an unconsidered issue or (in the best case) a simple testing of sensing application to the next generation of wireless sensors whose dependability is enabled through a seamless fault-injection-based validation campaign. Finally, comes the last question which is

When should the specified fault and related activities be performed ? A fault can be triggered either by the developer or depending on a particular condition. In the former case, the developer simply specifies when the injection should be performed. With the latter, an injection is triggered when a particular event, e.g., a message reception, arises at run time ; to support his mechanism, the monitoring system must be able to report the event. Note that a fault injector customized for distributed applications operating over wireless sensor networks should obviously not be based on the global state of the targeted system. This calls for providing a low intrusive and robust fault validation that keeps a local view of the overall system based on a in-network monitoring system ; post analysis is then applied to ensure that the proper fault insertion has been provided based on the local state. This low intrusiveness constraint, combined with low-resources constraints, result in in a system in which developing a fault-injection system is a really challenging task. These three questions, if successfully answered, will lead to the development of a successful fault injection campaign.

4. Conclusion

Wireless sensor networks hold great promise as an enabler of a wide spectrum of pervasive applications, including monitoring and geo-localisation, grounded upon data collection and signal treatment. However, as the range of applications extends in the fields of industrial and mission-critical, expectation drift towards dependability. Several learnfull campaigns demonstrated the difficulty inherent to the effective deployment and operation of wireless sensor networks under harsh conditions. Sensor networks are in fact much more exposed to natural perturbations, calibration problems and hardware/software failures. Assuming that these networks operate cheaply and that one may remedy to under performs by e.g., replacing, rebooting or deploying more embedded systems, does not scope with the reality : sensors are not currently cheap, and an increased number of sensors induces an increase in the complexity of the overall system. Meeting the challenging task of developing dependable applications built on top of robust sensor networks requires not only to provide fault-tolerant sensing and actuating capabilities but also to extensively test, evaluate and validate these capabilities. In this context, the lack of appropriate system for evaluating the dependability of the protocols and applications running across wireless sensor networks forces developers to conduct exhausting testing campaigns. Such *ad hoc* techniques are not sufficient to guarantee availability and reliability of the protocols and applications. To tackle this problem, we presente herein the premise of a software fault injector which allows developers to express faults related to signaling, communication and distributed processing. Considering the distributed nature of sensor networks, we focus on communication-related faults, by investigating for this purpose interception and manipulation of incoming and outgoing communication. Our main objective is to design a distributed fault injector to allow the user to test a wireless sensor network application without being physically present on the test bed. In our system, the code for performing tests will be seamlessly deployed and run over sensors nodes. Al-

though our work is only a first step towards this vision, we attempted to demonstrate its flexibility and usefulness through two case studies.

5. Bibliographie

Avizienis A., Laprie J., Randell B., « Dependability and its threats : a taxonomy », *IFIP Internation federation for information processing*, 2008.

Barrenetxea G., Ingelrest F., Schaefer G., Vetterli M., « The hitchhiker's guide to successful wireless sensor network deployment », *6th ACM Conference on embedded networked sensor systems*, p. 43-56, 2008.

Barrenetxea G., Ingelrest F., Schaefer G., Vetterli M., « AVR-INJECT : a tool for injecting faults in wireless sensor networks », *IEEE international Symposium on parallel and distributed processing (IPDPS)*, p. 1-8, 2009.

Boots M., Robin A., *Sensor Model Language (SensorML), v1.0.0.* 2007.

Borcea C., Intanagonwiwat C., Kang P., al., « Spatial Programming using spart messages : design and implementation », *International conference on distributed computing systems (ICDCS)*, 2004.

Chen Y., Gnawali O., Kazandjieva M., al., « Surviving Sensor Network Software Faults », *22nd ACM Symposium on Operating System Principles (SOSP)*, 2009.

Cox S., *Observations and measurements, v1.0.* 2007.

Dawson S., Jahanian F., Mitton T., « ORCHESTRA : a probing and fault injection environment for testing protocol implementations », *Computer Performance and Dependability Symposium, International*, vol. 0, p. 56, 1996.

Demaille A., Herault T., Peyronnet S., « Probabilistics verification of sensor networks », *4th IEEE International Conference on Computer Science, Research, Innovation and Vision for the Future (RIVF)*, 2006.

Dong J. S., Sun J., Taguchi K., al., « Model-based validation of QOS properties of biomedical sensor networks », *8th International Conference on Embedded Software*, 2008.

Dunkels A., Gronvall B., Voigt T., « Contiki - a lightweight and flexible operating system for Tiny Networked Sensors », *IEEE EMNETS*, 2004.

Fehnker A., Hoesel L., Mader A., « Modelling and verification of the LMAC protocol for wireless sensor networks », *6th international conference on integrated formal methods (IFM)*, 2007.

Girolami M., lenzi S., Furfari F., Chessa S., « SAIL : a Sensor Abstraction and Integration Layer for Context Awareness », *34th Euromico Control Conference Engineering and AdvancedS Applications*, p. 374-381, 2008.

Janakiram D., Venkateswarlu R., Nitin S., « COMiS : Component Oriented Middleware for Sensor Networks », *IEEE Workshop on Local Area and Metropolitan Networks (LANMAN)*, 2005.

Jung-Drebes R., Jacques-Silva G., da Trindade J. F., al., « A kernel-based communication fault injector for dependability testing of distributed systems », *International Haifa verification conference*, 2006.

Kim K., Mukhtar H., Joo S., S.Yoo, Park S. D., 6LoWPAN Management Information Base, Technical report, IETF Internet Draft, www.ietf.org, 2010.

Levis P., Gay D., Culler D., « Active sensor networks », *USENIX/ACM Symposium on networked systems designs and implementations (NSDI)*, 2005.

Milla K., Kish S., « A low cost micro processor and infrared sensor system for automating water infiltration measurements », *Computer and electronics in agriculture*, 2006.

Mourad M., Bertrand-Krajewski J., « Programming models for sensor networks : a survey », *ACM transactions on sensor networks (TOSN)*, 2008.

Mukhtar H., K. S. J., Schoenwaelder J., SNMP optimizations for 6LoWPAN, Technical report, IETF Internet Draft, www.ietf.org, 2009.

Ni K., Ramathan N., Chehade M., al., « Sensor network data fault type », *ACM Transaction on sensor networks*, 2009.

Sailhan F., Astic I., Michel F., Pitrey C., Uy M., Gressier-Soudan E., Gerbaud P., Forgeot H., « Sauvegarde du patrimoine, conception d'une solution de localisation et de surveillance à base de RFIDs actifs, défis et perspectives », *INFORSID-GEDSIP workshop*, 2009.

Thelen J., Goense D., Langendoen K., « Radio Wave Propagation in a Patatoe Field », *1th worshop on wireless network measurement*, 2005.

van Hoesel L., Havinga P., « A lightweight medium access protocol (LMAC) for wireless sensor networks : Reducing preamble transmissions and transceiver state switchesProbabilistics verification of sensor networks », *1st International Workshop on Networked Sensing Systems (INSS)*, 2004.

ANNEXE POUR LE SERVICE FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :

GEDSIP

2. AUTEURS :

Françoise Sailhan[*] *— Thierry Delot*[**] *— Animesh Pathak*[***] *— Aymeric Puech*[****] *— Matthieu Roy*[*****]

3. TITRE DE L'ARTICLE :

Dependable Wireless Sensor Networks

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :

Dependable Wireless Sensor Networks

5. DATE DE CETTE VERSION :

14 mars 2010

6. COORDONNÉES DES AUTEURS :

– adresse postale :

[*] CNAM, 292 rue Saint-Martin, F-75141 Paris. firstName.surname@cnam.fr

[**] Université de Valenciennes, Bat. ISTV2, Le Mont Houy 59313 Valenciennes. firstName.surname@univ-valenciennes.fr

[***] Projet Arles, INRIA Rocquencourt, domaine de Voluceau, BP 105, Le Chesnay. firstName.surname@inria.fr

[****] SmartGrains, Paris Innovation Bourse, 5 rue d'Uzès, 75002 Paris. firstName.surname@smartgrains.fr

[****] TSF Group, CNRS ; LAAS ; Université de Toulouse ; UPS , INSA , INP, ISAE ; 7 avenue du Colonel Roche, F-31077 Toulouse, France. surname@laas.fr

– téléphone : 00 00 00 00 00

– télécopie : 00 00 00 00 00

– e-mail : guillaume.laurent@ens2m.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :

LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :

Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
`http://www.revuesonline.com`